CLAIMS

What is claimed is:

1. A system, comprising:

a first processor;

a second processor coupled to the first processor;

an operating system that executes exclusively only on the first processor and not on the second processor; and

a middle layer software running on the first processor and that distributes tasks to run on either or both processors.

2. The system of claim 1 wherein the middle layer software comprises a Java virtual machine.

3. The system of claim 1 further comprising a synchronization unit coupled to the first and second processors, said synchronization unit synchronizes the execution of the first and second processors.

4. The system of claim 3 wherein the synchronization unit causes the first processor to transition to a wait mode while the second processor executes a task.

6. The system of claim 4 wherein the first processor is transitioned from the wait mode to a fully operational mode by a signal being asserted by the either the first or second processor to the synchronization unit.

7. The system of claim 1 further comprising a shared TLB containing a plurality of entries in which virtual-to-physical address translations are stored, each entry also containing a task ID field in which a task ID associated with the corresponding translation and with a task running on the first or second processor is stored.

8. The system of claim 7 wherein the operating system selectively flushes some of the entries in the shared TLB based on task ID.

9. The system of claim 7 wherein the middle layer software selectively flushes some of the entries in the shared TLB based on task ID.

10. The system of claim 9 wherein the middle layer software comprises a Java virtual machine.

11. The system of claim 7 wherein some of the shared TLB entries are invalidated, and those entries that are invalidated have task IDs that are associated with tasks that are running or have run on only one of the first or second processors.

12. The system of claim 1 wherein the second processor has a programmable context and autonomously switches its own context without support from the operating system executing on the first processor.

13. The system of claim 1 wherein the second processor includes a programmable task ID register which contains a value indicative of the task currently running on the second processor that is written by the middle layer software running on the first processor.

14. A method usable in a multi-processor system, comprising:

executing an operating system on only one of a plurality of processors; and

distributing tasks to each of the plurality of processors by middle layer software running on the processor on which the operating system executes.

15. The method of claim 14 wherein distributing tasks comprises distributing tasks by a Java virtual machine.

16. The method of claim 14 further comprising causing the processor on which the operating system executes to transition to a wait mode while another processor executes tasks and subsequently transitioning the processor in the wait mode to an active mode as a result of a signal being asserted by any of the plurality of processors.

17. The method of claim 14 wherein each task has a unique task identifier value and the method further comprises writing virtual-to-physical address translations and task identifier values associated with the task to which the translations pertain into a translation lookaside buffer that is shared between the plurality of processors.

18. The method of claim 17 further selecting task identifier values and invalidating entries in the translation lookaside buffer that contain the selected task identifier values and not invalidating other entries in the translation lookaside buffer.

19. The method of claim 14 wherein, in a processor having a context and that does not execute the operating system, autonomously switching said context without support from the operating system.

20. The method of claim 14 further comprising writing a task ID register by the processor executing the operating system, the task ID register contained in another processor.